Jurnal Penelitian Sains ; hal 45 - 60
No.2. Oktober 1997

ALGORITMA JUMLAH HIMPUNAN BEBAS MAKSIMUM PADA POHON

ERWIN
Jurusan Matematika FMIPA Universitas Sriwijaya

ABSTRAK

Penelitian ini bertujuan untuk membuat algoritma jumlah himpunan bebas maksimum pada pohon. Jumlah himpunan bebas maksimum pada pohon dengan n verteks adalah:

$$\mathbf{m(T)} = \begin{cases} 2^{k-1}+1; \textit{jika } n=2k \\ 2^{k}; \textit{ jika } n=2k+1 \end{cases}$$

PENDAHULUAN

Suatu graf terhubung yang tidak memiliki siklus dinamakan *pohon* T. (Kreyszig, Erwin, 1988). *Terhubung* berarti bahwa dari setiap verteks di dalam T ke verteks lain di dalam T selalu ada lintasan . Pohon termasuk jenis graf yang paling penting dan pohon dijumpai dalam berbagai penerapan. Pohon dapat digunakan untuk menunjukkan, mengorganisasikan, dan menganalisis Pohon termasuk jenis graf yang paling penting dan pohon dijumpai dalam berbagai penerapan. Pohon dapat digunakan untuk menunjukkan, mengorganisasikan, dan menganalisis jaringan listrik, hubungan produsen-konsumen dan relasi bisnis lainnya, informasi di dalam sistem basis data, struktur sintak program komputer, dan lain sebagainya.

Kebenaran *Teorema Empat Warna* (four color theorem) telah diduga sejak lama dan baru akhirnya terbuktikan setelah menggunakan *Bilangan Kromatik Rusuk* χ_e (G) dari graf G. Pembuktian teorema empat warna dapat dilihat pada Appel dan Haken (1976). Lebih lanjut, salah satu pengembangan dari teorema empat warna dibuat algoritma untuk menentukan bilangan kromatik dari suatu graf (E. Lawler,1976).

Algoritma tersebut memiliki kerumitan $O(mn(1+\sqrt[3]{3})^n)$. n dan m masing - masing adalah banyaknya verteks dan rusuk. Bilangan $\sqrt[3]{3}$ merupakan hasil dari teorema Moon dan Moser (1965) yang membahas graf dengan n verteks tidak dapat memliki lebih dari $3^{n/3}$ himpunan bebas maksimal. Algoritma yang dikembangkan oleh E Lawler (1976) dan jumlah himpuan bebas maksimum oleh Moon dan Moser (1965) merupakan pembahasan untuk graf yang tak terhubung. Sedangkan, rumusan dan algoritma untuk menentukan jumlah himpunan bebas maksimum pada graf terhubung dikembangkan oleh Griggs,Grinstead dan Guichar. Sementara itu, rumusan danalgoritma untuk menentukan jumlah himpunan bebas maksimum pada pohon perlu dibangun melalui penurunan yang telah ada.

HASIL DAN PEMBAHASAN

1. Pengertian Graf Terhubung dan Tak Terhubung.

Graf, dilambangkan dengan G = (V,E), terdiri atas dua himpunan terhingga V dan E Himpunan V bukan himpunan kosong dan unsur - unsurnya disebut *verteks*, sedangkan unsur - unsur himpuan E disebut *rusuk*, sedemikian sehingga setiap rusuk menghubungkan dua verteks yang dinamakan *verteks ujung* rusuk tersebut.

Graf G disebut terhubung jika untuk setiap dua verteks sebarang v dan w di G ada lintasan dari v ke w. Dalam hal lainnya, G disebut tak terhubung. Graf terhubung yang tidak memiliki siklus dinamakan pohon T (Kreyszig, E . 1988). Graf tak terhubung mempunyai paling sedikit dua graf bagian terhubung, masing - masing graf bagian ini disebut *komponen* graf tak terhubung itu.

2. Bilangan Kromatik

Wiitala,Stephen A (1985) mendefinisikan *bilangan kromatik* adalah jumlah minimum warna yang diperlukan untuk mewarnai graf sesuai dengan aturan tertentu. Bilangan kromatik pada

3. Himpunan Bebas Maksimum

Definisi T adalah pohon, V(T) adalah himpunan verteks dan n=| V(T)| adalah jumlah verteks. $S \subseteq V(T)$ adalah *himpunan bebas* jika tidak ada dua verteks dari S dihubungkan dengan rusuk pada T. S adalah *himpunan bebas maksimum* jika S adalah bebas dan setiap verteks dari V(T) - S dihubungkan oleh rusuk dengan paling sedikit satu verteks di S. m(T) adalah jumlah himpunan bebas maksimum di T (m(∅)=1).

T adalah pohon, V(T) adalah himpunan verteks dan n=| V(T)| adalah jumlah verteks. $S \subseteq V(T)$ adalah himpunan bebas jika tidak ada dua verteks dari S dihubungkan dengan rusuk pada T. S adalah himpunan bebas maksimum jika S adalah bebas dan setiap verteks dari V(T) - S dihubungkan oleh rusuk dengan paling sedikit satu verteks di S. m(T) adalah jumlah himpunan bebas maksimum di T (m(∅)=1). Untuk setiap pohon T dengan n simpul , jumlah maksimum m(T) adalah :

$$m(T) = \begin{cases} 2^{k-1}+1 & \textit{jika } n = 2k \\ 2^{k} & \textit{jika } n = 2k+1 \end{cases}$$

Algoritma Jumlah Himpunan Bebas Maksimum

Algoritma dibuat dengan kode semu, hal ini dilakukan untuk efisiensi algoritma. Analisis kompleksitas waktunya $O(n^2)$. Algoritma jumlah himpunan bebas maksimum pada pohon sebagai berikut :

```
                {Deklarasi Prosedur  PROGRAM HIMPUNAN BEBAS MAKSIMAL}
Procedure set matriks ke 0(ukuran_simpul )
Var Declaration I,J : byte ;
[Mulai]
For I := 1 to ukuran_simpul do { Inisialisasi matriks nxn dengan nilai elemen = 0}
   For J := 1 to ukuran_simpul do
   adj[I,J]<-- 0; RETURN

Procedure Simpul_awal(sSA)
[Mulai]
xGl[sSA ] <-- 160;                                  {Meletakkan simpul awal}
```

```
yGl[sSA] <---- 90;
outtextxy(xGl[sSA]-3,yGl[sSA]-4,'*');
RETURN
Procedure Gambar_simpul (sudutGs,saGs,sGS)
{ Menggambar tongkat panjang 1 dan simpul serta mengisi  elemen matriks ajasensi}
Var Declaration R,x,y  : byte;
[Mulai]
sudutGl[saGS,sGS] <---- sudutGs;

Repeat for <---- 1 to 20 do begin
          x <---- xGl[saGS]  + Round(R*cos(sudutGS));
          y <---- yGl[saGS]  + Round (R*sin(sudutGS));
putpixel(x,y,white);
[END Repeat]


xGl[sGS]  <----  xGl[saGS]  + Round(21*cos(sudutGS));
yGl[sGS]  <----  yGl[saGs]  + Round(21*sin(sudutGS));
outtextxy(xGl[sGS]-3, yGl[sGS]-4,'*');
adj[saGs,sGS] <---- 1;                          {Set elemen matrik adjensi = 1}
adj[sGs,saGs] <---- 1;
RETURN


Procedure Hapus_simpul( sHS,saHS : byte) { Menghapus dan merubah nilai elemen matriks
                                           ajasensi = 0 dari tongkat panjang 1 yang telah
                                           dibuat}
Var Declaration  R,x,y  : byte;  sudutHS : real
[Mulai]
x       <---- xGl[sHS];
y       <---- yGl[sHS];
uk      <---- imagesize(x-3,y-4,x+4,y+1);
getmem(gbr,uk);
getimage (x-3,y-4,x+4,y+1,gbr^);
putimage (x-3,y-4, gbr^,xorput);
freemem(gbr,uk);
sudutHS      <----  sudutGl[saHS,sHS] + pi;
Repeat  for R  <---- 1 to 18 do
x          <---- xGl[sHS]  + Round (R*cos(sudutHs));
y          <---- yGl[sHS]  + Round(R*sin(sudutHS));
uk         <---- imagesize(x-1,y-1,x+1,y+1);
getmem (gbr,uk);
getimage (x-1,y-1,x+1,y+1, gbr^);
putimage (x-1,y-1, gbr^, xorput);
freemem (gbr,uk)
END Repeat
```

```
adj[saHS,sHS] := 0;
adj[sHS,saHS] := 0;
outtextxy(xGl[saHS]-3,yGl[saHS]-4,'*');
RETURN

Procedure Tampilan_matriks(ukuran_simpul ) { Menampilkan matriks ajasensi dari pohon
Variabel Declaration  I,J : byte                ekstremal yang dipilih }
[Mulai]
 Write : 'matriks ajasensi '
Repeat for I <---  1 to n do
  For J <---  1 to n do
 IF I <---  1 THEN
 Write: chr(96+j)
      Write: chr(196)+chr(196
Write: adj[i,j]
END Repeat
    Write : chr(96+i)+chr(179)
 IF wherey >= 19 THEN
Write : Elemen matriks selanjutnya simetris
   i <---  n;
Write : Tekan sembarang tombol
Ch_matriks:=readkey;
setgraphmode(modegrafik);
 RETURN


Procedure Pilih_pohon;                  { Pilihan bentuk pohon }
[Mulai]
count_en <--- count_en+1;
IF count_en<>en THEN
settextstyle(2,0,4);
outtextxy : Bentuk Pohon Ini Akan Diproses (Y/N)
ch  <---  readkey;
settextstyle(0,0,1);
IF upcase(ch)='Y' THEN
CALL Tampilan_matriks(n); exit;
ELSE
settextstyle(2,0,4);
outtextxy : Semua Bentuk Pohon Sudah Ditampilkan
outtextxy : Tekan Sembarang Tombol
settextstyle(0,0,1);
ch:=readkey;
CALL Tampilan_matriks(n);
EXIT;
 uk:=imagesize(10,150,270,170);
 getmem(gbr,uk);
 getimage(10,150,270,170,gbr^);
```

```
 putimage (10,150,gbr^,xorput);
reemem(gbr,uk);
RETURN Procedure Tongkat_panjang_0 (ukuran_simpul){ Menggambar pohon ekstremal yang
dibentuk                                            oleh tongkat panjang 0}
Var Declaration  temcou,K,I  : byte;
[Mulai]
simpul_awal(1);
cou      <---      1;
k        <---      trunc((ukuran_simpul-1)/2).
Repeat  For I:= 1 to k do
inc(cou);
CALL Gambar_simpul(pi/2+2*pi*I/K,1,cou);
temcou           <--- cou;inc(cou);
CALL Gambar_simpul(pi/2+2*pi*I/K, temcou,cou);
END Repeat
settextstyle(2,0,4);
outtextxy :'Untuk jumlah simpul ganjil
outtextxy : hanya ada satu bentuk pohon
outtextxy : Tekan Sembarang Tombol
settextstyle(0,0,1);
Ch       <--- readkey;
CALL Tampilan_matriks(n);
IF upcase(ch)='Y' THEN exit;
RETURN

Procedure Tongkat_panjang_1 (ukuran_simpul )  { Menggambar pohon ekstremal yang dibentuk
                                                oleh tongkat panjang 1 }
Var  Declaration  temcou,I,J : byte;
[Mulai]
CALL Simpul_awal(1);
cou      <--- 2;
CALL Gambar_simpul(pi,1,cou);
k        <---      trunc((ukuran_simpul-2)/2);
Repeat For I:=1 to K do
inc(cou);
CALL Gambar_simpul(3*pi/2-pi*I/(K+1),2,cou);
temcou   <--- cou;
inc(cou);
CALL Gambar_simpul(3*pi/2-pi*I/(K+1),temcou,cou);
END Repeat
CALL Pilih_pohon;
IF upcase(ch)='Y' THEN exit;
J := trunc(K/2);
Repeat For I := 1 to J do
temcou := cou;
dec(cou);
CALL Hapus_simpul(temcou,cou);  gambar_simpul(pi*I/(K+1)-pi/2,1,temcou);
```

```
CALL Hapus_simpul(cou,2);
CALL Gambar_simpul(pi*I/(K+1)-pi/2,temcou,cou);
dec(cou);
IF count_en<en THEN
CALL Pilih_pohon;
        IF upcase(ch)='Y' THEN exit;END
END Repeat
RETURN
Procedure Tongkat_panjang_3(ukuran_simpul){ Menggambar pohon ekstremal yang dibentuk
                                            oleh tongkat panjang 3 }
Var Declaration temcou,I,J : byte
[Mulai]
IF ukuran_simpul <= 6 THEN exit
clearviewport
CALL set_matriks_ke_0(n)
CALL simpul_awal(1)
cou <---- 1
Repeat for I:= 1 to 3 do
temcou <---- cou;
inc(cou);
CALL gambar_simpul(pi,temcou , cou);
End Repeat
K <---- trunc((ukuran_simpul-4)/2);
Repeat For I := 1 to K-1 do
inc(cou);
CALL gambar_simpul(3*pi/2-pi*I/(K+1),4,cou);
        temcou <----cou;
inc(cou);
CALL gambar_simpul (3*pi/2-pi*I/(K+1), temcou,cou);
End Repeat
inc(cou);
temcou <---- cou;
CALL Gambar_simpul(pi/(K+1)-pi/2,1,temcou);
inc(cou);
    CALL Gambar_simpul(pi/(K+1)-pi/2,temcou,cou);
dec(cou,2);
CALL Pilih_pohon;
IF upcase(ch)='Y' THEN exit;
J <---- trunc(K/2);

Repat For I := 2 to J do
temcou := cou;
dec(cou);
CALL Hapus_simpul(temcou,cou);
CALL Gambar_simpul(pi*I/(K+1)-pi/2,1,temcou);
CALL Hapus_simpul(cou,4);
CALL Gambar_simpul(pi*I/(K+1)-pi/2,temcou,cou);
```

```
dec(cou);
End Repat
IF Count_en<en THEN
CALL Pilih_pohon;
IF upcase(ch)='Y' THEN exit;
RETURN
Function pangkatI (a,b )                    { Menghitung pangkat untuk bilangan integer }
Var Declaration i,pang : integer;
[Mulai]
Pang := 1;
For i:= 1 to b do
pang := pang * a;
pangkatI :=pang;
[Selesai]
Function pangkatII (a,b : real) : real;              { Menghitung pangkat untuk bilangan real }
[Mulai]
    pangkatII :=round(exp(ln(a)*b));
[Selesai]

Procedure hitung_MT;                        { Menghitung nilai MT dengan rumus }
[Mulai]
Repeat
Write : 'Banyak simpul (0<n<', ordo_max+1,'):'
Read  : n
IF (n<1) or (n>ordo_max) THEN
       Writeln : 'Data Anda Salah, Banyak Simpul Harus 0<n<',ordo_max+1,'):'
       Write   : 'Tekan Sembarang Tombol Untuk Ulangi Proses'
      ch :=readkey;
Until ((n>0) and (n<= ordo_max));
IF n mod 2 <--- 0 THEN
kgenap <--- n div 2;
en <--- kgenap-1;
mtgenap <---pangkatI(2,(kgenap-1))+1;
Writeln : 'Banyaknya Himp.Bebas Maks. Yang Dapat Dibentuk :'
Writeln : 'm(T) : ',MTgenap:2);
Writeln  : 'Banyaknya Pohon Ekstremal e(n):',end
ELSE
en <---1;
kganjil<---  (n-1)/2;
    MTganjil <--- pangkatII(2,kganjil);
    Writeln : 'Banyaknya Himp.Bebas Maks. Yang Dapat Dibentuk:'
    Writeln : 'm(T) : ',MTganjil
    Write : 'Tekan Sembarang Tombol'
RETURN

Procedure Inisialisasi;
Var Declaration baris,kolom : byte;
```

```
[Mulai]
Repaet For  baris := 1 to n do
Repeat For kolom := 1 to n do
S[baris,kolom]:=0;qmin[baris,kolom]:=0;
CALL Pilih_pohon;
IF upcase(ch)='Y' THEN exit;
RETURN
Function pangkatI (a,b )                    { Menghitung pangkat untuk bilangan integer }
Var Declaration  i,pang : integer;
[Mulai]
Pang := 1;
For i:= 1 to b do
pang := pang * a;
pangkatI :=pang;
[Selesai]


Function pangkatII (a,b : real) : real;              { Menghitung pangkat untuk bilangan real }
[Mulai]
    pangkatII :=round(exp(ln(a)*b));
[Selesai]

Procedure hitung_MT;                                 { Menghitung nilai MT dengan rumus }
[Mulai]
Repeat
Write : 'Banyak simpul (0<n<', ordo_max+1,'):'
Read  : n
IF (n<1) or (n>ordo_max) THEN
       Writeln : 'Data Anda Salah, Banyak Simpul Harus 0<n<',ordo_max+1,'):'
       Write    : 'Tekan Sembarang Tombol Untuk Ulangi Proses'
      ch :=readkey;
Until ((n>0) and (n<= ordo_max));
IF n mod 2 <--- 0 THEN
kgenap <--- n div 2;
en <--- kgenap-1;
mtgenap <---pangkatI(2,(kgenap-1))+1;
Writeln : 'Banyaknya Himp.Bebas Maks. Yang Dapat Dibentuk  :'
Writeln  : 'm(T) : ',MTgenap:2);
Writeln  : 'Banyaknya Pohon Ekstremal e(n):',en

ELSE
en <---1;
kganjil<---  (n-1)/2;
    MTganjil <--- pangkatII(2,kganjil);
    Writeln : 'Banyaknya Himp.Bebas Maks. Yang Dapat Dibentuk:'
    Writeln : 'm(T) : ',MTganjil
    Write : 'Tekan Sembarang Tombol'
```

```
RETURN

Procedure Inisialisasi
Var Declaration baris,kolom : byte;
[Mulai]
Repaet For  baris := 1 to n do
Repeat For kolom := 1 to n do
S[baris,kolom]:=0;qmin[baris,kolom]:=0;
RETURN


Procedure Langkah_forward
Var  Declaration j : byte
[Mulai]
   CALL Cari_X;
   For j :=1 to n do s[count+1,j]:=s[count,j];
     s[count+1,x]:=1;
     CALL isi_qmin;
     CALL isi_qplus;
     count :=count+1;
RETURN

Procedure Backtrack
[Mulai]
count :=count-1;
CALL cari_x;
s[count,x]:=0;
qplus[count,x]:=0;
qmin[count,x]:=1;
kond_fwd:=false;
RETURN

Procedure Cek_irisan (xbar:integer; var kondisi :boolean); { Mengecek irisan antara Gamma x
                                                            dengan qplus }
var Declaration xcol : integer;kond_gamma :boolean;
[Mulai]
kondisi :=true;

xcol :=1;
Repeat
Gamma(xbar,xcol,kond_gamma);
IF kond_gamma THEN
IF qplus[count,xcol]=1 THEN
kondisi :=false;
xcol:=xcol+1;
Until xcol>n;
RETURN
```

```
Procedure Cek_Elemen_Qmin (col:byte: var kondisi:boolean);   { Mengambil elemen qmin }
[Mulai]
kondisi :=false;
Repeat
IF qmin[count,col]=1 THEN
x:=col;
col:=n;


kondisi:=true;
col:=col+1; until col>n;
[Selesai]

Procedure Cek_qplus(var kondisi : boolean)                    { Mengecek qplus }
Var Declaration col : integer;
[Mulai]
col:=1;
kondisi :=true;
Repeat
        IF qplus[count,col]=0 THEN
          col :=col+1
        ELSE
          kondisi :=false;
    col:=n+1;
     Until col>n;
RETURN
Procedure Cek_qmin(var kondisi : boolean)                     { Mengecek Qmin }
Var Declaration col : integer;
[Mulai]
col:=1;
Set kondisi = true;
Repeat
   IF qmin[count,col]=0 THEN
      col:=col+1
  ELSE
      Set kondisi = false;
      col:=n+1;
    UNTIL col>n;
RETURN

Procedure Uji_1                 { Menguji apakah kondisi x elemen qmin sedemikian sehingga
                                  Gamma x barisan dengan q plus }
Var Declaration i: byte ;
[Mulai]]
CALL cek_elemen_qmin(1,kond_elemen);
IF kond_elemen THEN
CALL cek_Irisan(x,kond_irisan);
```

```
IF kond_irisan=false THEN
For i:= x to n do
CALL  cek_elemen_qmin(x+1,kond_elemen);
CALL cek_irisan(x,kond_irisan);
IF kond_irisan=true THEN i:=n;
ELSE kond_irisan :=false;
 IF  (kond_elemen and kond_irisan) THEN
   Set hasil_uji_1 := true

ELSE hasil_uji_1 := false;
RETURN
```

Procedure Uji_2A { Menguji apakah kondisi qplus dan qmin merupakan himpunan kosong }

```
[Mulai]
CALL Cek_qplus(kond_qplus);
CALL cek_qmin(kond_qmin);
IF (kond_qplus and kond_qmin) THEN
   Set hasil_uji_2A := true
ELSE hasil_uji_2A := false;
RETURN
```

Procedure Uji_2B; { Menguji apakah kondisi qplus merupakan himpunan kosong atau qmin tidak merupakan himpunan kosong }

```
[Mulai]
IF (not (kond_qplus) or kond_qmin) THEN
   Set hasil_uji_2B := true
ELSE hasil_uji_2B :=false;
RETURN
```

Procedure Isi_mis; { Memebentuk himpunan bebas maksimal }

```
Var Declaration j : byte;
[Mulai]
miscount := miscount+1;
For j:= 1 to n do

mis [miscount,j] := s[count,j];
Set kond_fwd :=false;

RETURN
```

Procedure Uji_K;

```
[Mulai]
IF count = 1 THEN
   CALL cek_qplus(kond_qplus);
IF kond_qplus THEN
count <-- 0;
```

```
RETURN

Procedure Output                    { Menampilkan himpunan bebas maksimal }
Var Declaration i,j : integer;mis_abjad : array [1..mis_max] of string[ordo_max];
Procedure Tampilan_mis;
Var  Declaration i : integer;
[Mulai]
  Writeln : 'Himpunan Bebas Maksimal Yang Dapat Dibentuk :'
  For i :=1 to miscount do
    Writeln('mis[',i,'] = ',mis_abjad[i]);

IF whereY >=20 THEN
    Writeln : 'Tekan Sembarang Tombol Untuk Melanjutkan'
        lanjut :=readkey;
   Writeln : 'Himpunan Bebas Maksimal Yang Dapat Dibentuk'
Writeln : 'Jumlah Himpunan Bebas Maksimal = ',miscount
RETURN


Procedure Cek_mis                   { Mengecek antara nilai MT dan miscount }
[Mulai]
IF (miscount = MTganjil) or (miscount = MTgenap) THEN
Writeln : 'Terbukti bahwa m(T) = mis = ',miscount
RETURN

{Mulai Algoritma Himpunan Bebas Maksimal}
Algoritma HIMPUNAN BEBAS MAKSIMAL. Mencari himpunan bebas maksimal beserrta grafiknya
untuk n data jumlah simpul.
Const Declaration
ordo_max = 20;
mis_max = 1000;
ghor = chr(205);
skrats = chr(201);
sknats = chr(187);
skrbwh = chr(200);
sknbwh = chr(188);
gver = chr(186);
Type Declaration
abj = array [1..ordo_max] of char;
Drivergrafik,modegrafik : integer;

Var Declaration
cou,k : byte ;
xGl,yGl : array [1..ordo_max]  of integer;
sudutGl : array [1..ordo_max,1..ordo_max] of real;
Ch,Ch_matriks : char;
slh,x,y : integer;
```

```
gbr : pointer;
uk : word;
count_en : byte;
n : byte;
adj : array [1..ordo_max,1..ordo_max] of byte;
abjad : string;
Mis : array [1..mis_max,1..ordo_max] of byte;
miscount : integer ;
MTgenap, kgenap, en : integer ;
MTganjil, kganjil : real ;lanjut : char;

[Inisialisasi]
CountEn ← 0
[Hitung Nilai MT]
    Repeat
  Write : 'Banyak Simpul'
  Read : N
  IF N < 1 or N > Ordo Maks THEN                    {Periksa banyak simpul}
Write : 'Data Salah'
Write : 'Proses diulang'
Until N > 0 dan N < = Ordo Maks
              IF N mod 2 ← 0 THEN                   {Jika N genap}
                Kgenap ← N div 2
                En ← Kgenap - 1
                Mtgenap ← Pangkat + I(2,(Kgenap - 1))+1
            Write : 'Banyaknya Himpunan Bebas Maksimal yang dapat dibentuk'
            Write : 'MTGenap'
            Write : 'Banyak Pohon Ekstermal',En
              ELSE
                En ← 1                              {Jika N ganjil}
                Kganjil ← PangkatII(2,KGanjil)
            Write : 'MTGanjil'
[Set Matrik ke Nol]
    CALL Set_Matrik_ke_0;

   IF N ← Round (N/2) THEN                          {Bulatkan nilai n ke nilai terdekat}
      CALL Tongkat_Panjang_1(n)
       IF N >=8 THEN
          CALL Tongkat_Panjang_3(n)
       ELSE
          CALL Tongkat_Panjang_0(n
IF Kond_Fwd THEN
      CALL Prosedur Lankah_Forward;
      CALL Prosedure Uji_1;
   IF NOT Hasil_Uji_1 THEN
      CALL Prosedur Uji_2A;
      IF NOT Hasil_Uji_2A THEN
```

DAFTAR PUSTAKA

A. Wiitala, Stephen.(1985). **Discrete Mathematics A Unified Approach**. Mc Graw-Hill Book Company. New York.

C.L Liu. (1985). Element of Discrete Mathematics. Mc Graw-Hill, Inc. New York

D. Cohen. (1984). **Counting Stable Sets in Trees**. Institute de Recherche Mathematique Avancee Pub. Strasbourg.Prancis.

E. Lawler. (1976). **A Note on The Complexity of The Chromatic Number Problem**. Inform. Proc Lett, 5.

F. Harary. (1969). **Graph Theory**. Addison-Wisley, Reading,MA.

Jean-Paul Tremblay and Paul G Soreson.(1984). **An Introduction To Data Structure With Applications** 2/e.MC Graw-Hill International Editions.Singapore

J.Griggs, C. Grinstead and D. Guichard, **The Number of Maximal Independent Sets in a Connected Graph,** Preprint.

J. Moon And L. Moser . (1965). **On Cliques in Graphs**. Israel J. Math,3

Mark Allen Weiss, **Data Structures and Algoritme Analysis in C**. Benjamin/Cuming Publishing Company,Inc

Kreyszig, Erwin. (1988). **Advaced Engineering Mathematics 6th Ed**. John Wiley & Sons Inc.New York.

Seymour Lipschutz,(1986). **Theory and Problems of Data Structures**. Mc Graw-Hill International Editions.Singapore.